CLAIMS:

1. A system for implementing a function between a first processor and a second processor, said system comprising:

a shared virtual memory (SVM) coupled to the first and second processors, said SVM to store one or more double-ended queues (Deques);

one or more execution units (EUs) of the first processor, a first of said one or more EUs associated with a first of the one or more Deques, to push the function to said first Deque; and

one or more request handler threads executing on said second processor, at least a first of the one or more request handler threads to:

pop the function from the first Deque;

execute the function; and

generate a completion signal to said first EU in response to completion of the function.

2. The system of claim 1, wherein the function is a callback request comprising:

a function pointer to specify said function to be executed by said second processor; and

an argument pointer to specify a location in said SVM for data to be provided to said executed function and for results generated by said executed function to be provided to said first EU.

3. The system of claim 2, wherein said first request handler thread is further to sleep in response to failure to pop the callback request from any of said one or more Deques.

4. The system of claim 3, wherein said sleep comprises execution of a Monitor/Mwait instruction.

5. The system of claim 1, wherein said first request handler thread is further to pop the function from a second Deque, said second Deque associated with a second EU, in response to detecting that said first Deque is empty.

6. The system of claim 1, wherein said one or more Deques are pinned in said SVM to prevent page faults in response to access attempts by said one or more EUs and said one or more request handler threads.

7. The system of claim 1, wherein said completion signal is to set a completion flag in said SVM and said first EU is further to poll said completion flag.

8. The system of claim 1, wherein said completion signal is to trigger a continuation kernel on said first EU, said continuation kernel to restore previously saved stack variables.

9. The system of claim 1, wherein said first processor is a Graphics Processor Unit (GPU) and said second processor is a general purpose Central Processor Unit (CPU).

10. A method for function operation between a first processor and a second processor, said method comprising:

pushing a function, by a first execution unit (EU) of one or more EUs of said first processor, to a first double-ended queue (Deque) of one or more Deques, said first Deque stored in a shared virtual memory (SVM) coupled to said first and second processors;

popping the function from said first Deque, by a first request handler thread of one or more request handler threads executing on said second processor;

executing, by said first request handler thread, the function; and

generating, by said first request handler thread, a completion signal to said first EU in response to completion of said function.

11. The method of claim 10, wherein the function is a callback request comprising:

a function pointer to specify said function to be executed by said second processor; and

an argument pointer to specify a location in said SVM for data to be provided to said executed function and for results generated by said executed function to be provided to said first EU.

12. The method of claim 11, further comprising sleeping, by said first request handler thread, in response to a failure to pop the callback request from any of said Deques.

13. The method of claim 12, wherein said sleeping further comprises execution of a Monitor/Mwait instruction.

14. The method of claim 10, further comprising popping, by said first request handler thread, the function from a second Deque, said second Deque associated with a second EU, in response to detecting that said first Deque is empty.

15. The method of claim 10, further comprising pinning said one or more Deques in said SVM to prevent page faults in response to access attempts by said one or more EUs and said one or more request handler threads.

16. The method of claim 10, wherein generating said completion signal further comprises setting a completion flag in said SVM for polling by said first EU.

17. The method of claim 10, wherein generating said completion signal further comprises triggering a continuation kernel on said first EU, said continuation kernel to restore previously saved stack variables.

18. At least one computer-readable storage medium having instructions stored thereon which when executed by a processor result in the following operations for a function between a first processor and a second processor, said operations comprising:

pushing the function, by a first execution unit (EU) of one or more EUs of said first processor, to a first double-ended queue (Deque) of one or more Deques, said one or more Deques stored in a shared virtual memory (SVM) coupled to said first and second processors;

popping the function from said first Deque, by a first request handler thread of one or more request handler threads executing on said second processor;

executing, by said first request handler thread, the function; and

generating, by said first request handler thread, a completion signal to said first EU in response to completion of said function.

19. The computer-readable storage medium of claim 18, wherein the function is a callback request comprising:

a function pointer to specify said function to be executed by said second processor; and

an argument pointer to specify a location in said SVM for data to be provided to said executed function and for results generated by said executed function to be provided to said first EU.

20. The computer-readable storage medium of claim 19, further comprising sleeping, by said first request handler thread, in response to a failure to pop the callback request from any of said one or more Deques.

21. The computer-readable storage medium of claim 20, wherein said sleeping further comprises execution of a Monitor/Mwait instruction.

22. The computer-readable storage medium of claim 18, further comprising popping, by said first request handler thread, the function from a second Deque, said second Deque associated with a second EU, in response to detecting that said first Deque is empty.

23. The computer-readable storage medium of claim 18, further comprising pinning said one or more Deques in said SVM to prevent page faults in response to access attempts by said one or more EUs and said one or more request handler threads.

24. The computer-readable storage medium of claim 18, wherein generating said completion signal further comprises setting a completion flag in said SVM for polling by said first EU.

25. The computer-readable storage medium of claim 18, wherein generating said completion signal further comprises triggering a continuation kernel on said first EU, said continuation kernel to restore previously saved stack variables.